


```
FFFFFFFFF 000000 RRRRRRRR UU UU DDDDDDDD FFFFFFFFFF WW WW LL
FFFFFFFFF 000000 RRRRRRRR UU UU DDDDDDDD FFFFFFFFFF WW WW LL
FF 00 00 RR RR UU UU DD DD FF WW WW LL
FF 00 00 RR RR UU UU DD DD FF WW WW LL
FF 00 00 RR RR UU UU DD DD FF WW WW LL
FFFFFFFF 00 00 RRRRRRRR UU UU DD DD FFFFFFFF WW WW LL
FFFFFFFF 00 00 RRRRRRRR UU UU DD DD FFFFFFFF WW WW LL
FF 00 00 RR RR UU UU DD DD FF WW WW LL
FF 00 00 RR RR UU UU DD DD FF WW WW LL
FF 00 00 RR RR UU UU DD DD FF WW WW LL
FF 00 00 RR RR UU UU DD DD FF WW WW LL
FF 000000 RR RR UUUUUUUUU DDDDDDDD FF WW WW LLLLLLLLLL
FF 000000 RR RR UUUUUUUUU DDDDDDDD FF WW WW LLLLLLLLLL
```

```
LL I I I I I S S S S S S S
LL I I I I I S S S S S S
LL I I S S
LL I I S S
LL I I S S
LL I I S S
LL I I S S S S S S
LL I I S S S S S
LL I I S S
LL I I S S
LL I I S S
LL I I S S
LLLLLLLLLL I I I I I S S S S S S
LLLLLLLLLL I I I I I S S S S S S
```



```
1 0001 0 MODULE FOR$$UDF_WL (%TITLE'FORTRAN Write List Directed UDF'  
2 0002 0 IDENT = '1-028' ! File: FORUDFWL.B32 Edit: SBL1028  
3 0003 0 ) =  
4 0004 1 BEGIN  
5 0005 1  
6 0006 1  
7 0007 1 *****  
8 0008 1 *  
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *  
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *  
11 0011 1 * ALL RIGHTS RESERVED. *  
12 0012 1 *  
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *  
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *  
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *  
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *  
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *  
18 0018 1 * TRANSFERRED. *  
19 0019 1 *  
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *  
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *  
22 0022 1 * CORPORATION. *  
23 0023 1 *  
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *  
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *  
26 0026 1 *  
27 0027 1 *  
28 0028 1 *****  
29 0029 1  
30 0030 1 ++  
31 0031 1 FACILITY: FORTRAN Support Library - not user callable  
32 0032 1  
33 0033 1 ABSTRACT:  
34 0034 1  
35 0035 1 ENVIRONMENT: User access mode; reentrant AST level or not.  
36 0036 1  
37 0037 1 AUTHOR: Jonathan M. Taylor; CREATION DATE: 27-Jul-77  
38 0038 1  
39 0039 1 MODIFIED BY:  
40 0040 1 Jonathan M. Taylor, 27-Jul-77: Version 0  
41 0041 1 Steven B. Lionel, VAX/VMS V2.0  
42 0042 1  
43 0043 1 Previous edit history deleted. SBL 20-Oct-1980  
44 0044 1 1-001 - Update version number and copyright notice. JBS 16-NOV-78  
45 0045 1 1-002 - Put value in impossible case of CASE statement to  
46 0046 1 keep BLISS compiler happy. JBS 27-NOV-78  
47 0047 1 1-003 - Change REQUIRE file names from FOR... to OTS... JBS 07-DEC-78  
48 0048 1 1-004 - Change ISB$A BUF_PTR, BUF_END, BUF_HIGH, BUF_BEG to LUB. DGP 08-Jan-79  
49 0049 1 1-005 - Use 32-bit addresses for externals. JBS 27-JAN-1979  
50 0050 1 1-006 - Add G, H, DC, GC support. SBL 14-Mar-1979  
51 0051 1 1-007 - Fix FC so that it doesnt access second longword of arg! SBL 14-Mar-79  
52 0052 1 1-008 - Add new integer output routines. SBL 26-Mar-79  
53 0053 1 1-009 - Check for record overflow. Make complex conform to ANSI  
54 0054 1 standard. SBL 30-Mar-1979  
55 0055 1 1-010 - Have D and G values only print 15 fraction digits. SBL 18-Apr-79  
56 0056 1 1-011 - Complete H floating support. SBL 12-Jun-1979  
57 0057 1 1-012 - REC level no longer puts in a first blank. Put them in here.
```



```

: 58      0058 1 | SBL 26-Jun-1979
: 59      0059 1 | 1-013 - Put D and G back the way they were. 1PG25.16. SBL 28-Jun-1979
: 60      0060 1 | 1-014 - OT$CVT_x_TG routines are now FOR$CVT_x_TG. SBL 3-JUL-79
: 61      0061 1 | 1-015 - If not sequential org, give "mixed file access modes". SBL 3-Oct-1979
: 62      0062 1 | 1-016 - Make 1-015 read sequential access. SBL 4-Oct-1979
: 63      0063 1 | 1-017 - Insert leading blank for continued character strings. SBL 4-Oct-1979
: 64      0064 1 | 1-018 - Remove access check, done in FOR$IO BEG. SBL 5-Dec-1979
: 65      0065 1 | 1-019 - Add DO WRITE to do REC1 calls through dispatch table so that
: 66      0066 1 | FOR$UDF WN (NAMELIST) can call UDF_WL1. SBL 20-Oct-1980
: 67      0067 1 | 1-020 - Add REPEAT CNT parameter to UDF_WL1 for NAMELIST. SBL 29-Oct-1980
: 68      0068 1 | 1-021 - Signal FOR$OUTSTAOVE immediately rather than storing in ISB. SBL 7-Nov-1980
: 69      0069 1 | 1-022 - Make leading blank and repeat count work correctly for complex. SBL 10-Dec-1980
: 70      0070 1 | 1-023 - Don't separate character values with a space, since the standard
: 71      0071 1 | says not to. Add BU datatype to be same as B. SBL 9-June-1981
: 72      0072 1 | 1-024 - Specify digits in exp as 3 for G floating and 4 for H floating
: 73      0073 1 | since editing is implicitly Ew.dEe and the exponent letter
: 74      0074 1 | therefore cannot be dropped, per standard. JAW 25-Aug-1981
: 75      0075 1 | 1-025 - Start a new record for any character string that won't fit in
: 76      0076 1 | the current (partly-filled) record. JAW 26-Aug-1981
: 77      0077 1 | 1-026 - Finish edit 1-024, which should have increased the field size
: 78      0078 1 | used for a complex G floating value by one. JAW 31-Aug-1981
: 79      0079 1 | 1-027 - Remove extra leading space from D and G output and reduce
: 80      0080 1 | the number of fraction digits for G to 15. SBL 10-Nov-1981
: 81      0081 1 | ***** - VMS V3.0
: 82      0082 1 | 1-028 - JSB to REC-level routines through dispatch tables to allow
: 83      0083 1 | list-directed internal files. Use prologue file. SBL 21-Apr-1983
: 84      0084 1 | --
```



```

86      0085 1 |
87      0086 1 | PROLOGUE FILE:
88      0087 1 |
89      0088 1 |
90      0089 1 REQUIRE 'RTLIN:FORPROLOG';           ! FORTRAN definitions
91      0155 1 |
92      0156 1 |
93      0157 1 | TABLE OF CONTENTS:
94      0158 1 |
95      0159 1 |
96      0160 1 FORWARD ROUTINE
97      0161 1     FOR$$UDF_WL0: JSB_UDF0 NOVALUE, ! UDF initialization
98      0162 1     FOR$$UDF_WL1: CALL_CCB NOVALUE, ! format one user I/O list element
99      0163 1     FOR$$UDF_WL9: JSB_UDF9 NOVALUE, ! UDF termination
100     0164 1     DO_WRITE: JSB_RECT NOVALUE;    ! Call the REC level to write a record
101     0165 1 |
102     0166 1 |
103     0167 1 | EQUATED SYMBOLS:
104     0168 1 |
105     0169 1 |     NONE
106     0170 1 |
107     0171 1 |
108     0172 1 | OWN STORAGE:
109     0173 1 |
110     0174 1 |     None
111     0175 1 |
112     0176 1 | EXTERNAL REFERENCES:
113     0177 1 |
114     0178 1 |
115     0179 1 EXTERNAL ROUTINE
116     0180 1 |
117     0181 1     FOR$$SIGNAL_STO : NOVALUE,           ! Signal fatal error
118     0182 1 |
119     0183 1 |     The following are general library routines available for any
120     0184 1 |     one's use (value is true if fits in field):
121     0185 1 |
122     0186 1     OT$$CVT_L_TL,           ! Convert long to text, L format
123     0187 1     OT$$CVT_L_TI,           ! Convert long to text, I format
124     0188 1     FOR$CVT_F_TG,           ! Convert F to text, G format
125     0189 1     FOR$CVT_D_TG,           ! Convert D to text, G format
126     0190 1     FOR$CVT_G_TG,           ! Convert G to text, G format
127     0191 1     FOR$CVT_H_TG;           ! Convert H to text, G format
128     0192 1 |
129     0193 1 |
130     0194 1 EXTERNAL
131     0195 1     FOR$$AA_REC_PRO: VECTOR,           ! PIC arrays of record processor procedures
132     0196 1     FOR$$AA_REC_PR1: VECTOR,           ! Write a record in REC level of
133     0197 1     FOR$$AA_REC_PR9: VECTOR;           ! abstraction. Indexed by I/O statement
134     0198 1           ! type (ISB$B_STTM_TYPE)

```

```
: 136      0199 1 GLOBAL ROUTINE FOR$$UDF_WLO
: 137      0200 1 : JSB_UDFO NOVA CUE =
: 138      0201 1
: 139      0202 1 ++
: 140      0203 1 FUNCTIONAL DESCRIPTION:
: 141      0204 1
: 142      0205 1 Perform UDF level write list-directed I/O initialization.
: 143      0206 1 Call record level processor to initialize buffer pointers.
: 144      0207 1
: 145      0208 1 CALLING SEQUENCE:
: 146      0209 1
: 147      0210 1 JSB FOR$$UDF_WLO ( )
: 148      0211 1
: 149      0212 1 FORMAL PARAMETERS:
: 150      0213 1
: 151      0214 1
: 152      0215 1 IMPLICIT INPUTS:
: 153      0216 1
: 154      0217 1 CCB Pointer to current logical unit block (LUB)
: 155      0218 1
: 156      0219 1 IMPLICIT OUTPUTS:
: 157      0220 1
: 158      0221 1 LUB$A_BUF_BEG set to start of buffer
: 159      0222 1
: 160      0223 1 ROUTINE VALUE:
: 161      0224 1 COMPLETION CODES:
: 162      0225 1
: 163      0226 1 NONE
: 164      0227 1
: 165      0228 1 SIDE EFFECTS:
: 166      0229 1
: 167      0230 1 FOR$_MIXFILACC if file is not sequential access.
: 168      0231 1
: 169      0232 1 --
: 170      0233 2 BEGIN
: 171      0234 2
: 172      0235 2 EXTERNAL REGISTER
: 173      0236 2 CCB : REF $FOR$CCB_DECL;
: 174      0237 2
: 175      0238 2 JSB_RECO (FOR$$AA_REC_PRO + .FOR$$AA_REC_PRO [.CCB [ISB$B_STTM_TYPE] -
: 176      0239 2 ISB$K_FORSTTY[0 + 1]);
: 177      0240 2 CCB [LUB$A_BUF_BEG] = .CCB [LUB$A_BUF_PTR]; ! Beginning of buffer
: 178      0241 2
: 179      0242 1 END;
```

```
.TITLE FOR$$UDF_WL FORTRAN Write List Directed UDF
.IDENT \1-028\
```

```
.EXTRN FOR$$SIGNAL_STO
.EXTRN OTSS$CVT_L_TC, OTSS$CVT_L_TI
.EXTRN FOR$CVT_F-TG, FOR$CVT_D-TG
.EXTRN FOR$CVT_G-TG, FOR$CVT_H-TG
.EXTRN FOR$$AA_REC_PRO
.EXTRN FOR$$AA_REC_PRI
.EXTRN FOR$$AA_REC_PRI9
```


FOR\$\$UDF_WL
1-028

FORTRAN Write List Directed UDF

L 14
16-Sep-1984 00:52:40
14-Sep-1984 12:32:54

VAX-11 Bliss-32 V4.0-742
[FORRTL.SRC]FORUDFWL.B32;1

Page 5
(3)

.PSECT _FOR\$CODE,NOWRT, SHR, PIC,2

50	FF71	CB	9A	00000	FOR\$\$UDF WLO::		
					MOVZBL	-143(CCB), R0	: 0239
50	00000000G0040	D0	00005		MOVL	FOR\$\$AA_REC_PRO[R0], R0	: 0238
	00000000G0040	16	0000D		JSB	FOR\$\$AA_REC_PRO[R0]	: 0240
BC	AB	B0	AB	D0	00014	MOVL	-80(CCB), -88(CCB)
				05	00019	RSB	: 0242

; Routine Size: 26 bytes, Routine Base: _FOR\$CODE + 0000


```
181 0243 1 GLOBAL ROUTINE FOR$$UDF_WL1 (  
182 0244 1     ELEM_TYPE,  
183 0245 1     ELEM_SIZE,  
184 0246 1     ELEM_ADR,  
185 0247 1     FC_FLAG,  
186 0248 1     REPEAT_CNT)  
187 0249 1     : CALL_CCB NOVALUE =  
188 0250 1  
189 0251 1 ++  
190 0252 1 FUNCTIONAL DESCRIPTION:  
191 0253 1  
192 0254 1     Write list-directed User Data Formatter.  
193 0255 1     Accept an I/O element, format it, and put it in the record buffer.  
194 0256 1     Calls record level processors to perform the actual I/O.  
195 0257 1  
196 0258 1 CALLING SEQUENCE:  
197 0259 1  
198 0260 1     CALL FOR$$UDF_WL1 (elem_type.rlu.v, elem_size.rlu.v, elem_adr.rx.r  
199 0261 1     [, fc_flag.rlu.v [, repeat_cnt.rl.v]])  
200 0262 1  
201 0263 1 FORMAL PARAMETERS:  
202 0264 1  
203 0265 1     ELEM_TYPE.rlu.v      Type code of user I/O list  
204 0266 1                        element. Form: ELEM_TYPE x  
205 0267 1                        x = B,W,L,BU,WU,LU,F,D,FC or T.  
206 0268 1     ELEM_SIZE.rlu.v    Size of user I/O list element  
207 0269 1                        in addressable machine units  
208 0270 1                        If 0, this is end of I/O list call.  
209 0271 1     ELEM_ADR.rx.r        Adr. of user I/O list element  
210 0272 1                        x = b, w, l, bu, wu, lu, f, d, fc, t,  
211 0273 1                        g, h, dc or gc.  
212 0274 1     [FC_FLAG.rlu.v]    If present:  
213 0275 1                        0 - real part of COMPLEX number  
214 0276 1                        1 - imaginary part of COMPLEX number  
215 0277 1                        2 - not complex number, but repeat cnt present  
216 0278 1     [REPEAT_CNT.rl.v]    If present, the value written is to prefaced by  
217 0279 1                        a repeat count of the form n*. The value of the  
218 0280 1                        parameter is the repeat count.  
219 0281 1  
220 0282 1 IMPLICIT INPUTS:  
221 0283 1  
222 0284 1     NONE  
223 0285 1  
224 0286 1 IMPLICIT OUTPUTS:  
225 0287 1  
226 0288 1     NONE  
227 0289 1  
228 0290 1 ROUTINE VALUE:  
229 0291 1 COMPLETION CODES:  
230 0292 1  
231 0293 1     NONE  
232 0294 1  
233 0295 1 SIDE EFFECTS:  
234 0296 1  
235 0297 1     NONE  
236 0298 1  
237 0299 1 --
```



```
238 0300 1
239 0301 2 BEGIN
240 0302 2
241 0303 2 EXTERNAL REGISTER
242 0304 2 CCB: REF $FOR$CCB_DECL;
243 0305 2
244 0306 2 BUILTIN
245 0307 2 ACTUALCOUNT;
246 0308 2
247 0309 2 BIND
248 0310 2 FIELD SIZE =
249 0311 2 UPLIT BYTE(
250 0312 2 0, ! not used
251 0313 2 0, ! not used
252 0314 2 5, ! BU (same as B for FORTRAN)
253 0315 2 2, ! WU
254 0316 2 2, ! LU
255 0317 2 0, ! not used
256 0318 2 5, ! B
257 0319 2 7, ! W
258 0320 2 12, ! L
259 0321 2 0, ! not used
260 0322 2 15, ! F
261 0323 2 24, ! D
262 0324 2 11, ! FC - absolute minimum for real part
263 0325 2 20, ! DC - absolute minimum for real part
264 0326 2 ! types between DC and G not used
265 0327 2 24, ! G
266 0328 2 43, ! H
267 0329 2 19) ! GC - absolute minimum for real part
268 0330 2 :VECTOR[, BYTE];
269 0331 2 MAP
270 0332 2 ELEM_ADR: REF VECTOR; ! element is call-by-reference
271 0333 2
272 0334 2 LOCAL
273 0335 2 DIFF, ! number of bytes left in record buffer
274 0336 2 LELEM_TYPE, ! If first part of COMPLEX then FC else .ELEM_TYPE
275 0337 2 REPEAT_COUNT, ! Local copy of repeat count
276 0338 2 REPEAT_DSC: DSC$DESCRIPTOR, ! Descriptor for repeat string
277 0339 2 REPEAT_STR: VECTOR [12, BYTE], ! Repeat count string
278 0340 2 L; ! Field length of this element
279 0341 2
280 0342 2 !+
281 0343 2 ! If we're being called to write the first part of a COMPLEX number,
282 0344 2 ! then change the ELEM_TYPE to COMPLEX. If called for the second part
283 0345 2 ! of a COMPLEX number then just return since the first part really
284 0346 2 ! wrote both parts...
285 0347 2 !-
286 0348 2
287 0349 2 LELEM_TYPE =
288 0350 3 (IF ACTUALCOUNT() GTR (FC_FLAG - ELEM_TYPE)/%UPVAL
289 0351 3 THEN
290 0352 4 BEGIN
291 0353 4 IF .FC_FLAG EQL 1
292 0354 4 THEN
293 0355 4 RETURN;
294 0356 4 IF .FC_FLAG LSS 1
```



```

295      0357 4      THEN
296      0358 4          SELECTONE .ELEM_TYPE OF
297      0359 4              SET
298      0360 4                  [DSC$K_DTYPE_F]: DSC$K_DTYPE_FC;
299      0361 4                  [DSC$K_DTYPE_D]: DSC$K_DTYPE_DC;
300      0362 4                  [DSC$K_DTYPE_G]: DSC$K_DTYPE_GC;
301      0363 4              TES
302      0364 4      ELSE
303      0365 4          .ELEM_TYPE
304      0366 4      END
305      0367 3      ELSE
306      0368 2          .ELEM_TYPE);
307      0369 2
308      0370 2      !+
309      0371 2      !- compute field length
310      0372 2
311      0373 2      L = (IF .LELEM_TYPE EQL DSC$K_DTYPE_T
312      0374 3          THEN
313      0375 3              2 ! 1 character + leading blank minimum
314      0376 3          ELSE
315      0377 3              IF .LELEM_TYPE GEQU DSC$K_DTYPE_G
316      0378 3                  THEN
317      0379 3                      .FIELD_SIZE[.LELEM_TYPE - (DSC$K_DTYPE_G-DSC$K_DTYPE_DC-1)]
318      0380 3                  ELSE
319      0381 3                      .FIELD_SIZE[.LELEM_TYPE]);
320      0382 2
321      0383 2      !+
322      0384 2      !- Construct repeat count string.
323      0385 2
324      0386 2
325      0387 2      REPEAT_DSC [DSC$W_LENGTH] = 0;
326      0388 2      IF ACTUALCOUNT () GTR (REPEAT_CNT - ELEM_TYPE)/%UPVAL
327      0389 2      THEN
328      0390 2          BEGIN
329      0391 3              IF .REPEAT_CNT GTR 1
330      0392 3              THEN
331      0393 3                  BEGIN
332      0394 4                      LOCAL
333      0395 4                          FAO_DSC: DSC$DESCRIPTOR; ! For FAO control string
334      0396 4                          REPEAT_DSC [DSC$A_POINTER] = REPEAT_STR;
335      0397 4                          REPEAT_DSC [DSC$W_LENGTH] = 12;
336      0398 4                          REPEAT_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
337      0399 4                          REPEAT_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
338      0400 4                          FAO_DSC [DSC$A_POINTER] = UPLIT BYTE ('!SL*');
339      0401 4                          FAO_DSC [DSC$W_LENGTH] = %CHARCOUNT ('!SL*');
340      0402 4                          FAO_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
341      0403 4                          FAO_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
342      0404 4                          REPEAT_COUNT = .REPEAT_CNT;
343      0405 4                          $FAO (FAO_DSC,
344      0406 4                              REPEAT_DSC [DSC$W_LENGTH], ! Control string descriptor
345      0407 4                              REPEAT_DSC, ! Return length here
346      0408 4                              .REPEAT_COUNT); ! Output string descriptor
347      0409 4                              ! Value
348      0410 3                  END;
349      0411 2      END;
350      0412 2
351      0413 2      !+
```



```

352      0414 2      ! If there is not enough room in this record buffer for the element,
353      0415 2      ! then write this buffer and start anew.  If entire record wont
354      0416 2      ! fit value, return error FOR_OUTSTAOVE.
355      0417 2      !-
356      0418 2
357      0419 2      IF (.L + .REPEAT_DSC [DSC$W_LENGTH]) GTR CH$DIFF (.CCB[LUB$A_BUF_END], .CCB[LUB$A_BUF_PTR])
358      0420 2      THEN
359      0421 2          IF (.L + .REPEAT_DSC [DSC$W_LENGTH]) GTR .CCB [LUB$W_RBUF_SIZE]
360      0422 2              THEN
361      0423 2                  BEGIN
362      0424 2                      FOR$$SIGNAL_STO (FOR$K_OUTSTAOVE);
363      0425 2                      RETURN;
364      0426 2                      END
365      0427 2              ELSE
366      0428 2                  DO_WRITE ();
367      0429 2
368      0430 2
369      0431 2      !+
370      0432 2      ! If not complex, write leading space and repeat count.
371      0433 2      !-
372      0434 2
373      0435 2      IF .ELEM_TYPE EQL .LELEM_TYPE      ! True if not COMPLEX
374      0436 2      THEN
375      0437 2          BEGIN
376      0438 2              !+
377      0439 2              ! Don't put the leading space if this is a CHARACTER value unless
378      0440 2              ! it's also the beginning of the record.
379      0441 2              !-
380      0442 2              IF (.ELEM_TYPE NEQU DSC$K_DTYPE_T) OR
381      0443 2                  (.CCB [LUB$A_BUF_PTR] EQLA .CCB [LUB$A_BUF_BEG])
382      0444 2              THEN
383      0445 2                  CH$WCHAR A ('C' , CCB [LUB$A_BUF_PTR]);
384      0446 2                  CCB [LUB$A_BUF_PTR] = CH$MOVE (.REPEAT_DSC [DSC$W_LENGTH],
385      0447 2                      .REPEAT_DSC [DSC$A_POINTER], .CCB [LUB$A_BUF_PTR]);
386      0448 2              END;
387      0449 2
388      0450 2      !+
389      0451 2      ! If the element is a string literal, then move it into the record.
390      0452 2      !-
391      0453 2
392      0454 2      IF .LELEM_TYPE EQL DSC$K_DTYPE_T
393      0455 2      THEN
394      0456 2          BEGIN
395      0457 2              LOCAL
396      0458 2                  P;
397      0459 2                      ! Pointer to the string literal
398      0460 2              P = CH$PTR (.ELEM_ADR);
399      0461 2              L = .ELEM_SIZE;
400      0462 2
401      0463 2              !+
402      0464 2              ! If the string won't fit in the remaining portion of the
403      0465 2              ! buffer, and the buffer is not empty, write out the buffer and
404      0466 2              ! start a new record.
405      0467 2              !-
406      0468 2
407      0469 2              IF .L GTR CH$DIFF (.CCB [LUB$A_BUF_END], .CCB [LUB$A_BUF_PTR])
408      0470 2                  AND .CCB [LUB$A_BUF_PTR] NEQ .CCB [LUB$A_RBUF_ADR] + 1
```



```

      THEN
      BEGIN
      DO WRITE ();
      CH$WCHAR_A (%C' ', CCB [LUB$A_BUF_PTR]);
      END;

      !+
      ! If the string is longer than the record buffer,
      ! move in the part that fits and write the record.
      !-

      WHILE .L GTR (DIFF = CH$DIFF (.CCB[LUB$A_BUF_END], .CCB[LUB$A_BUF_PTR]))
      DO
      BEGIN
      CCB[LUB$A_BUF_PTR] = CH$MOVE (.DIFF, .P, .CCB[LUB$A_BUF_PTR]);
      P = CH$PLUS (.P, .DIFF);
      L = .L - .DIFF;
      IF .L GTR 0
      THEN
      BEGIN
      DO WRITE ();
      CH$WCHAR_A (%C' ', CCB [LUB$A_BUF_PTR]);
      END;
      END;

      !+
      ! move in the rest of the string.
      !-

      CCB[LUB$A_BUF_PTR] = CH$MOVE (.L, .P, .CCB[LUB$A_BUF_PTR]);
      END

ELSE
BEGIN
LOCAL
  DSC: BLOCK[8, BYTE];          ! static string descriptor for output field

  !+
  ! Perform the appropriate conversions.
  !-

  IF .ELEM_TYPE NEQU .LELEM_TYPE ! Only happens if item is complex
  THEN
  BEGIN
  LOCAL
    VALUE_ADDR,          ! Address of value
    STRING1: VECTOR [23, BYTE], ! Result of conversions
    STRING2: VECTOR [23, BYTE],
    LENGTH1,             ! Length of values
    LENGTH2,
    RPT_LENGTH,          ! Length of repeat count
    LEFT1,
    LEFT2,
    CONVERT_RTN,         ! Address of convert routine
```



```

: 466      0528 4      DIGITS,          ! digits_in_fraction
: 467      0529 4      EXP DIGITS;      ! digits_in_exp
: 468      0530 4      DSC [DSC$A_POINTER] = STRING1;
: 469      0531 4      SELECTONE .LELEM_TYPE OF
: 470      0532 4      SET
: 471      0533 4      [DSC$K_DTYPE_FC]:
: 472      0534 5      BEGIN
: 473      0535 5      DIGITS = 7;
: 474      0536 5      EXP DIGITS = 2;
: 475      0537 5      CONVERT RTN = FOR$CVT_F_TG;
: 476      0538 5      DSC [DSC$W_LENGTH] = T4;
: 477      0539 4      END;
: 478      0540 4      [DSC$K_DTYPE_DC]:
: 479      0541 5      BEGIN
: 480      0542 5      DIGITS = 16;
: 481      0543 5      EXP DIGITS = 2;
: 482      0544 5      CONVERT RTN = FOR$CVT_D_TG;
: 483      0545 5      DSC [DSC$W_LENGTH] = 23;
: 484      0546 4      END;
: 485      0547 4      [DSC$K_DTYPE_GC]:
: 486      0548 5      BEGIN
: 487      0549 5      DIGITS = 15;
: 488      0550 5      EXP DIGITS = 3;
: 489      0551 5      CONVERT RTN = FOR$CVT_G_TG;
: 490      0552 5      DSC [DSC$W_LENGTH] = 23;
: 491      0553 4      END;
: 492      0554 4      TES;
: 493      0555 4      IF NOT (.CONVERT_RTN) (.ELEM_ADR, DSC, .DIGITS, 0, 1,
: 494      0556 4      .EXP_DIGITS)
: 495      0557 4      THEN
: 496      0558 4      CCB [ISB$B_ERR_NO] = FOR$K_OUTCONERR;
: 497      0559 4
: 498      0560 4      !+
: 499      0561 4      ! Get length of real part.
: 500      0562 4      !-
: 501      0563 5      BEGIN
: 502      0564 5      LOCAL
: 503      0565 5      RIGHT; ! Boundary columns of converted value
: 504      0566 5      LEFT1 = CH$FIND_NOT_CH (.DSC [DSC$W_LENGTH], .DSC [DSC$A_POINTER], %C' ');
: 505      0567 5      RIGHT = CH$FIND_CH ? (.DSC [DSC$W_LENGTH] - CH$DIFF (.LEFT1, .DSC [DSC$A_POINTER])), .LEFT1, %C'
: 506      0568 5      IF CH$FAIL (.RIGHT)
: 507      0569 5      THEN
: 508      0570 5      RIGHT = CH$PLUS (.DSC [DSC$A_POINTER], .DSC [DSC$W_LENGTH]);
: 509      0571 5      LENGTH1 = CH$DIFF (.RIGHT, .LEFT1);
: 510      0572 4      END;
: 511      0573 4      IF .LELEM_TYPE EQL DSC$K_DTYPE_FC
: 512      0574 4      THEN
: 513      0575 4      VALUE_ADDR = ELEM_ADR [1]
: 514      0576 4      ELSE
: 515      0577 4      VALUE_ADDR = ELEM_ADR [2];
: 516      0578 4      DSC [DSC$A_POINTER] = STRING2;
: 517      0579 4      IF NOT (.CONVERT_RTN) (.VALUE_ADDR, DSC, .DIGITS, 0, 1,
: 518      0580 4      .EXP_DIGITS)
: 519      0581 4      THEN
: 520      0582 4      CCB [ISB$B_ERR_NO] = FOR$K_OUTCONERR;
: 521      0583 4
: 522      0584 4      !+

```



```

: 523      0585 4      ! Get length of imaginary part.
: 524      0586 4      !-
: 525      0587 5      BEGIN
: 526      0588 5      LOCAL
: 527      0589 5      RIGHT; ! Boundary columns of converted value
: 528      0590 5      LEFT2 = CH$FIND_NOT_CH (.DSC [DSC$W_LENGTH], .DSC [DSC$A_POINTER], %C' ');
: 529      0591 5      RIGHT = CH$FIND_CH ? (.DSC [DSC$W_LENGTH] - CH$DIFF (.LEFT2, .DSC [DSC$A_POINTER])), .LEFT2, %C'
: 530      0592 5      IF CH$FAIL (.RIGHT)
: 531      0593 5      THEN
: 532      0594 5      RIGHT = CH$PLUS (.DSC [DSC$A_POINTER], .DSC [DSC$W_LENGTH]);
: 533      0595 5      LENGTH2 = CH$DIFF (.RIGHT, .LEFT2);
: 534      0596 4      END;
: 535      0597 4
: 536      0598 4      !+
: 537      0599 4      ! If entire constant will fit on this line, put it there.
: 538      0600 4      ! If it won't go on this line, but will go on a new line,
: 539      0601 4      ! put entire constant on new line.
: 540      0602 4      ! Otherwise, if it can be split, split it. Error if either
: 541      0603 4      ! part is larger than record buffer size.
: 542      0604 4      !-
: 543      0605 4
: 544      0606 4      RPT_LENGTH = .REPEAT DSC [DSC$W_LENGTH];
: 545      0607 4      IF ? .LENGTH1 + .LENGTH2 + .RPT_LENGTH + 3) LSSP .CCB [LUB$W_RBUF_SIZE]
: 546      0608 4      THEN
: 547      0609 5      BEGIN
: 548      0610 5      IF (.LENGTH1 + .LENGTH2 + .RPT_LENGTH + 3) GTRP CH$DIFF (.CCB [LUB$A_BUF_END],
: 549      0611 5      .CCB [LUB$A_BUF_PTR])
: 550      0612 5      THEN
: 551      0613 5      DO WRITE (,);
: 552      0614 5      CH$WCHAR A (%C' ', CCB [LUB$A_BUF_PTR]);
: 553      0615 5      CCB [LUB$A_BUF_PTR] = CH$MOVE (.RPT_LENGTH,
: 554      0616 5      .REPEAT DSC [DSC$A_POINTER], CCB [LUB$A_BUF_PTR]);
: 555      0617 5      CH$WCHAR A (%C'(', CCB [LUB$A_BUF_PTR]);
: 556      0618 5      CCB [LUB$A_BUF_PTR] = CH$MOVE (.LENGTH1, .LEFT1, CCB [LUB$A_BUF_PTR]);
: 557      0619 5      CH$WCHAR A (%C' ', CCB [LUB$A_BUF_PTR]);
: 558      0620 5      CCB [LUB$A_BUF_PTR] = CH$MOVE (.LENGTH2, .LEFT2, CCB [LUB$A_BUF_PTR]);
: 559      0621 5      CH$WCHAR A (%C')', CCB [LUB$A_BUF_PTR]);
: 560      0622 5      END
: 561      0623 4      ELSE
: 562      0624 5      BEGIN
: 563      0625 5      IF (.LENGTH1 + .RPT_LENGTH + 2) GEQP .CCB [LUB$W_RBUF_SIZE]
: 564      0626 5      THEN
: 565      0627 6      BEGIN
: 566      0628 6      FOR$SIGNAL_STO (FOR$K_OUTSTAOVE);
: 567      0629 6      RETURN;
: 568      0630 5      END;
: 569      0631 5      IF (.LENGTH1 + .RPT_LENGTH + 2) GTRP CH$DIFF (.CCB [LUB$A_BUF_END],
: 570      0632 5      .CCB [LUB$A_BUF_PTR])
: 571      0633 5      THEN
: 572      0634 5      DO WRITE (,);
: 573      0635 5      CH$WCHAR A (%C' ', CCB [LUB$A_BUF_PTR]);
: 574      0636 5      CCB [LUB$A_BUF_PTR] = CH$MOVE (.RPT_LENGTH,
: 575      0637 5      .REPEAT DSC [DSC$A_POINTER], CCB [LUB$A_BUF_PTR]);
: 576      0638 5      CH$WCHAR A (%C'(', CCB [LUB$A_BUF_PTR]);
: 577      0639 5      CCB [LUB$A_BUF_PTR] = CH$MOVE (.LENGTH1, .LEFT1, CCB [LUB$A_BUF_PTR]);
: 578      0640 5      CH$WCHAR A (%C' ', CCB [LUB$A_BUF_PTR]);
: 579      0641 5      IF (.LENGTH2 + 2) GTRP .CCB [LUB$W_RBUF_SIZE]
```



```

580      0642  5      THEN
581      0643  6      BEGIN
582      0644  6      FOR$SIGNAL_STO (FOR$K_OUTSTAOVE);
583      0645  6      RETURN;
584      0646  5      END;
585      0647  5      DO WRITE ();
586      0648  5      CH$WCHAR A (%C' ', CCB [LUB$A_BUF_PTR]);
587      0649  5      CCB [LUB$A_BUF_PTR] = CH$MOVE (.LENGTH2, .LEFT2, .CCB [LUB$A_BUF_PTR]);
588      0650  5      CH$WCHAR_A (%C' '), CCB [LUB$A_BUF_PTR];
589      0651  4      END;
590      0652  4      END
591      0653  4      ELSE
592      0654  3      BEGIN
593      0655  3      DSC[DSC$W_LENGTH] = .L - 1;
594      0656  4      DSC[DSC$A_POINTER] = .CCB[LUB$A_BUF_PTR];
595      0657  4      IF NOT (
596      0658  4      CASE .LELEM_TYPE FROM DSC$K_DTYPE_BU TO DSC$K_DTYPE_D OF
597      0659  4      SET
600      0661  5      [ DSC$K_DTYPE_WU, DSC$K_DTYPE_LU]:
601      0662  5      OTSS$VT_L_TL (.ELEM_ADR, DSC);
602      0663  5      [ DSC$K_DTYPE_BU, DSC$K_DTYPE_B, DSC$K_DTYPE_W, DSC$K_DTYPE_I]:
603      0664  5      OTSS$VT_L_TI (.ELEM_ADR, DSC, T, .ELEM_SIZE);
604      0665  5      [ DSC$K_DTYPE_F, DSC$K_DTYPE_D]:
605      0666  5      BEGIN
606      0667  5      LOCAL
607      0668  5      D_VALUE: VECTOR[2]; ! holds double precision floating value
608      0669  5      D_VALUE[0] = .ELEM_ADR[0];
609      0670  5      D_VALUE[1] = (IF .ELEM_SIZE EQL %UPVAL
610      0671  6      THEN
611      0672  6      0
612      0673  6      ELSE
613      0674  6      .ELEM_ADR[1]);
614      0675  7      FOR$CVT_D TG(D_VALUE, DSC,
615      0676  7      (IF .ELEM_TYPE EQL DSC$K_DTYPE_F
616      0677  7      THEN
617      0678  7      7
618      0679  6      ELSE
619      0680  6      16), 1)
620      0681  7      END;
621      0682  7      [ INRANGE ]: 0;
622      0683  7      ! this can not happen
623      0684  6      [ OUTRANGE ]:
624      0685  5      CASE .LELEM_TYPE FROM DSC$K_DTYPE_G TO DSC$K_DTYPE_H OF
625      0686  5      SET
626      0687  5      [DSC$K_DTYPE_G]:
627      0688  5      FOR$CVT_G TG (ELEM_ADR[0], DSC, 15, 0, 1, 3);
628      0689  5      !-1 digit in integer part, 3 in exponent
629      0690  5      [DSC$K_DTYPE_H]:
630      0691  5      FOR$CVT_H TG (ELEM_ADR[0], DSC, 33, 0, 1, 4);
631      0692  5      !-1 digit in integer part, 4 in exponent.
632      0693  5
633      0694  5
634      0695  5
635      0696  5
636      0697  5
637      0698  5
```



```

: 637      0699 5      TES
: 638      0700 5      TES)
: 639      0701 4      THEN
: 640      0702 4
: 641      0703 4
: 642      0704 4      CCB[ISB$B_ERR_NO] = FOR$K_OUTCONERR;
: 643      0705 4      CCB[LUB$A_BUF_PTR] = CH$PLUS(CCB[LUB$A_BUF_PTR], .L - 1);
: 644      0706 4
: 645      0707 4      !+
: 646      0708 4      !- If there was a repeat count, left justify the value.
: 647      0709 4
: 648      0710 4
: 649      0711 4      IF .REPEAT_DSC [DSC$W_LENGTH] GTR 0
: 650      0712 4      THEN
: 651      0713 5          BEGIN
: 652      0714 5          LOCAL
: 653      0715 5          POS;
: 654      0716 5          POS = CH$FIND NOT CH (.DSC [DSC$W_LENGTH], .DSC [DSC$A_POINTER], %C' ');
: 655      0717 5          IF NOT CH$FAIC (.POS)
: 656      0718 5          THEN
: 657      0719 5              CCB [LUB$A_BUF_PTR] = CH$MOVE (CH$DIFF (.CCB [LUB$A_BUF_PTR], .POS),
: 658      0720 5                  .POS, .DSC [DSC$A_POINTER]);
: 659      0721 4          END;
: 660      0722 4      END;
: 661      0723 3      END;
: 662      0724 3
: 663      0725 2      END;
: 664      0726 2      RETURN;
: 665      0727 1      END;
```

```

18 14 0B 18 0F 00 0C 07 05 00 02 02 05 00 00 0001A P.AAA: .BYTE 0, 0, 5, 2, 2, 0, 5, 7, 12, 0, 15, 24, -
                                13 2B 00029 11, 20, 24, 43, 19
                                2A 4C 53 21 0002B P.AAB: .ASCII \!SL*\
                                FIELD SIZE= P.AAA
                                .EXTRN SYSS$FAO
                                07FC 00000 .ENTRY FOR$$UDF_WL1, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 0243
5E AC AE 9E 00002 MOVAB -84(SP), SP
58 04 AC D0 00006 MOVL ELEM_TYPE, R8
03 6C 91 0000A CMPB (AP), #3
01 2F 1B 0000D BLEQU 5$
10 AC D1 0000F CMPL FC_FLAG, #1
01 01 12 00013 BNEQ 1$
10 AC D5 00016 1$: TSTL FC_FLAG
23 14 00019 BGTR 5$
0A 58 D1 0001B CMPL R8, #10
05 05 12 0001E BNEQ 2$
56 0C D0 00020 MOVL #12, R6
1C 11 00023 BRB 6$
0B 58 D1 00025 2$: CMPL R8, #11
05 05 12 00028 BNEQ 3$
56 0D D0 0002A MOVL #13, R6
```


				12	11	0002D	BRB	6\$						
				58	D1	0002F	3\$: CMPL	R8, #27			0362			
				05	13	00032	BEQL	4\$						
				01	CE	00034	MNEGL	#1, R6						
				08	11	00037	BRB	6\$						
				1D	D0	00039	4\$: MOVL	#29, R6						
				03	11	0003C	BRB	6\$			0358			
				58	D0	0003E	5\$: MOVL	R8, LELEM_TYPE			0368			
				59	D4	00041	6\$: CLRL	R9			0374			
				56	D1	00043	CMPL	LELEM_TYPE, #14						
				07	12	00046	BNEQ	7\$						
				59	D6	00048	INCL	R9						
				02	D0	0004A	MOVL	#2, L						
				11	11	0004D	BRB	9\$						
				56	D1	0004F	7\$: CMPL	LELEM_TYPE, #27			0378			
				07	1F	00052	BLSSU	8\$						
				86	AF	46	9A	00054	MOVZBL	FIELD_SIZE-13[LELEM_TYPE], L	0380			
				05	11	00059	BRB	9\$						
				8C	AF	46	9A	0005B	8\$: MOVZBL	FIELD_SIZE[LELEM_TYPE], L	0382			
				4C	AE	B4	00060	9\$: CLRW	REPEAT_DSC	0388				
				04	6C	91	00063	CMPB	(AP), #4	0389				
				37	1B	00066	BLEQU	10\$						
				01	14	AC	D1	00068	CMPL	REPEAT_CNT, #1	0392			
				31	15	0006C	BLEQ	10\$						
				50	AE	40	AE	9E	0006E	MOVAB	REPEAT_STR, REPEAT_DSC+4	0397		
				4C	AE	010E000C	8F	D0	00073	MOVL	#17694732, REPEAT_DSC	0398		
				3C	AE	FF7D	CF	9E	0007B	MOVAB	P.AAB, FAO_DSC+4	0401		
				38	AE	010E0004	8F	D0	00081	MOVL	#17694724, FAO_DSC	0402		
				50	14	AC	D0	00089	MOVL	REPEAT_CNT, REPEAT_COUNT	0405			
				50	DD	0008D	PUSHL	REPEAT_COUNT		0409				
				50	AE	9F	0008F	PUSHAB	REPEAT_DSC					
				54	AE	9F	00092	PUSHAB	REPEAT_DSC					
				44	AE	9F	00095	PUSHAB	FAO_DSC					
				04	FB	00098	CALLS	#4, -SYS\$FAO						
				4C	AE	3C	0009F	10\$: MOVZWL	REPEAT_DSC, R1		0419			
				57	C0	000A3	ADDL2	L, R1						
				50	B4	AB	80	AB	C3	000A6	SUBL3	-80(CCB), -76(CCB), R0		
				50	D1	000AC	CMPL	R1, R0						
				0E	15	000AF	BLEQ	12\$						
				51	D2	AB	10	00	ED	000B1	CMPZV	#0, #16, -46(CCB), R1	0421	
				03	18	000B7	BGEQ	11\$						
				0220	31	000B9	BRW	38\$						
				0000V	30	000BC	11\$: BSBW	DO_WRITE			0428			
				56	D1	000BF	12\$: CMPL	R8, LELEM_TYPE			0435			
				1E	12	000C2	BNEQ	15\$						
				0E	D1	000C4	CMPL	R8, #14			0442			
				07	12	000C7	BNEQ	13\$						
				BC	AB	80	AB	D1	000C9	CMPL	-80(CCB), -68(CCB)	0443		
				07	12	000CE	BNEQ	14\$						
				B0	BB	80	20	90	000D0	13\$: MOVB	#32, a-80(CCB)	0445		
				B0	AB	D6	000D4	INCL	-80(CCB)					
				B0	BB	50	BE	4C	AE	28	000D7	14\$: MOV3	REPEAT_DSC, aREPEAT_DSC+4, a-80(CCB)	0447
				B0	AB	53	D0	000DE	MOVL	R3, -80(CCB)				
				4E	B0	AB	9E	000E2	15\$: MOVAB	-80(CCB), R10		0469		
				59	0C	AC	D0	000E9	BLBC	R9, 19\$				
				57	08	AC	D0	000ED	MOVL	ELEM_ADR, P		0460		
									MOVL	ELEM_SIZE, L		0461		

50	B4	AB	6A	C3	000F1	SUBL3	(R10), -76(CCB), R0	0469	
		50	57	D1	000F6	CMPL	L, R0		
			14	15	000F9	BLEQ	17\$		
50	EC	AB	01	C1	000FB	ADDL3	#1, -20(CCB), R0	0470	
		50	6A	D1	00100	CMPL	(R10), R0		
			0A	13	00103	BEQL	17\$		
			0000V	30	00105	BSBW	DO WRITE	0473	
	B0	BB	20	90	00108	MOVB	#32, a-80(CCB)	0474	
			AB	D6	0010C	INCL	-80(CCB)		
58	B4	AB	AB	C3	0010F	SUBL3	-80(CCB), -76(CCB), DIFF	0482	
		58	57	D1	00115	CMPL	L, DIFF		
			13	15	00118	BLEQ	18\$		
B0	BB		58	28	0011A	MOVC3	DIFF, (P), a-80(CCB)	0485	
		69	53	D0	0011F	MOVL	R3, -80(CCB)		
	B0	AB	58	C0	00123	ADDL2	DIFF, P	0486	
		59	58	C2	00126	SUBL2	DIFF, L	0487	
		57	E4	15	00129	BLEQ	17\$	0488	
			D8	11	0012B	BRB	16\$	0491	
B0	BB		57	28	0012D	MOVC3	L, (P), a-80(CCB)	0500	
		69	53	D0	00132	MOVL	R3, -80(CCB)		
	B0	AB		04	00136	RET		0454	
		56	58	D1	00137	CMPL	R8, LELEM_TYPE	0515	
			03	12	0013A	BNEQ	20\$		
			01C6	31	0013C	BRW	40\$		
	3C	AE	20	AE	9E	0013F	MOVAB	STRING1, DSC+4	0530
		0C		56	D1	00144	CMPL	LELEM_TYPE, #12	0533
				13	12	00147	BNEQ	21\$	
		53		07	D0	00149	MOVL	#7, DIGITS	0535
		52		02	D0	0014C	MOVL	#2, EXP_DIGITS	0536
		54	00000000G	00	9E	0014F	MOVAB	FOR\$CVT_F_TG, CONVERT_RTN	0537
	38	AE		0E	B0	00156	MOVW	#14, DSC	0538
				2A	11	0015A	BRB	24\$	0531
		0D		56	D1	0015C	CMPL	LELEM_TYPE, #13	0540
				0F	12	0015F	BNEQ	22\$	
		53		10	D0	00161	MOVL	#16, DIGITS	0542
		52		02	D0	00164	MOVL	#2, EXP_DIGITS	0543
		54	00000000G	00	9E	00167	MOVAB	FOR\$CVT_D_TG, CONVERT_RTN	0544
				12	11	0016E	BRB	23\$	0545
		1D		56	D1	00170	CMPL	LELEM_TYPE, #29	0547
				11	12	00173	BNEQ	24\$	
		53		0F	D0	00175	MOVL	#15, DIGITS	0549
		52		03	D0	00178	MOVL	#3, EXP_DIGITS	0550
		54	00000000G	00	9E	0017B	MOVAB	FOR\$CVT_G_TG, CONVERT_RTN	0551
	38	AE		17	B0	00182	MOVW	#23, DSC	0552
				52	DD	00186	PUSHL	EXP_DIGITS	0556
				01	DD	00188	PUSHL	#1	0555
				7E	D4	0018A	CLRL	-(SP)	
				53	DD	0018C	PUSHL	DIGITS	
			48	AE	9F	0018E	PUSHAB	DSC	
			0C	AC	DD	00191	PUSHL	ELEM_ADR	
		64		06	FB	00194	CALLS	#6, T(CONVERT_RTN)	
		05		50	E8	00197	BLBS	R0, 25\$	
		CB		3F	90	0019A	MOVB	#63, -144(CCB)	0558
3C	BE	FF70		20	3B	0019F	SKPC	#32, DSC, aDSC+4	0566
		38		02	12	001A5	BNEQ	26\$	
				51	D4	001A7	CLRL	R1	
		04		51	D0	001A9	MOVL	R1, LEFT1	

50	3C	AE	04	AE	C3	001AD	SUBL3	LEFT1, DSC+4, R0	0567
		51	38	AE	3C	001B3	MOVZWL	DSC, R1	
04	BE	50		51	C0	001B7	ADDL2	R1, R0	
		50		20	3A	001BA	LOCC	#32, R0, @LEFT1	
				02	12	001BF	BNEQ	27\$	
				51	D4	001C1	CLRL	R1	
				51	D5	001C3	TSTL	RIGHT	0568
				08	12	001C5	BNEQ	28\$	
		51	38	AE	3C	001C7	MOVZWL	DSC, RIGHT	0570
		51	3C	AE	C0	001CB	ADDL2	DSC+4, RIGHT	
58		51	04	AE	C3	001CF	SUBL3	LEFT1, RIGHT, LENGTH1	0571
		OC		56	D1	001D4	CMPL	LELEM_TYPE, #12	0573
				07	12	001D7	BNEQ	29\$	
50	OC	AC		04	C1	001D9	ADDL3	#4, ELEM_ADR, VALUE_ADR	0575
				05	11	001DE	BRB	30\$	
50	OC	AC		08	C1	001E0	ADDL3	#8, ELEM_ADR, VALUE_ADR	0577
	3C	AE	08	AE	9E	001E5	MOVAB	STRING2, DSC+4	0578
				52	DD	001EA	PUSHL	EXP_DIGITS	0580
				01	DD	001EC	PUSHL	#1	0579
				7E	D4	001EE	CLRL	-(SP)	
				53	DD	001F0	PUSHL	DIGITS	
			48	AE	9F	001F2	PUSHAB	DSC	
				50	DD	001F5	PUSHL	VALUE_ADR	
		64		06	FB	001F7	CALLS	#6, (CONVERT_RTN)	
		05		50	E8	001FA	BLBS	R0, 31\$	
		CB		3F	90	001FD	MOVB	#63, -144(CCB)	0582
3C	BE	FF70		20	3B	00202	SKPC	#32, DSC, @DSC+4	0590
		38		02	12	00208	BNEQ	32\$	
				51	D4	0020A	CLRL	R1	
				51	D0	0020C	MOVL	R1, LEFT2	
50	3C	6E		6E	C3	0020F	SUBL3	LEFT2, DSC+4, R0	0591
		AE	38	AE	3C	00214	MOVZWL	DSC, R1	
		51		51	C0	00218	ADDL2	R1, R0	
00	BE	50		20	3A	0021B	LOCC	#32, R0, @LEFT2	
		50		02	12	00220	BNEQ	33\$	
				51	D4	00222	CLRL	R1	
				51	D5	00224	TSTL	RIGHT	0592
				08	12	00226	BNEQ	34\$	
		51	38	AE	3C	00228	MOVZWL	DSC, RIGHT	0594
		51	3C	AE	C0	0022C	ADDL2	DSC+4, RIGHT	
56		51		6E	C3	00230	SUBL3	LEFT2, RIGHT, LENGTH2	0595
		59	4C	AE	3C	00234	MOVZWL	REPEAT DSC, RPT_LENGTH	0606
50		58		56	C1	00238	ADDL3	LENGTH2, LENGTH1, R0	0607
		51	03	A940	9E	0023C	MOVAB	3(RPT_LENGTH)[R0], R1	
51	D2	AB		00	ED	00241	CMPZV	#0, #T6, -46(CCB), R1	
		10		45	1B	00247	BLEQU	36\$	
				6A	C3	00249	SUBL3	(R10), -76(CCB), R0	0611
50	B4	AB		51	D1	0024E	CMPL	R1, R0	
		50		03	1B	00251	BLEQU	35\$	
				0000V	30	00253	BSBW	DO WRITE	0613
		57	B0	AB	9E	00256	MOVAB	-80(CCB), R7	0614
		B7		20	90	0025A	MOVB	#32, @0(R7)	
				67	D6	0025E	INCL	(R7)	
00	B7	50		59	28	00260	MOVC3	RPT_LENGTH, @REPEAT_DSC+4, @0(R7)	0616
		67		53	D0	00266	MOVL	R3, (R7)	
		B7	00	28	90	00269	MOVB	#40, @0(R7)	0617
				67	D6	0026D	INCL	(R7)	

00	B7	04	BE	58	28	0026F	MOV C3	LENGTH1, @LEFT1, @0(R7)	0618	
		67		53	D0	00275	MOVL	R3, (R7)		
		00	B7	2C	90	00278	MOVB	#44, @0(R7)	0619	
				67	D6	0027C	INCL	(R7)		
00	B7	00	BE	56	28	0027E	MOV C3	LENGTH2, @LEFT2, @0(R7)	0620	
		67		53	D0	00284	MOVL	R3, (R7)		
		00	B7	29	90	00287	MOVB	#41, @0(R7)	0621	
				67	D6	0028B	INCL	(R7)		
				04	0028D		RET		0607	
52	D2	AB	52	02	A948	9E 0028E	36\$:	MOVAB	2(RPT_LENGTH)[LENGTH1], R2	0625
		10		00	ED	00293		CMPZV	#0, #T6, -46(CCB), R2	
				41	1B	00299		BLEQU	38\$	
		50	B4	6A	C3	0029B		SUBL3	(R10), -76(CCB), R0	0632
			50	52	D1	002A0		CPL	R2, R0	
				03	1B	002A3		BLEQU	37\$	
				0000V	30	002A5		BSBW	DO WRITE	0634
			57	B0	AB	9E 002A8	37\$:	MOVAB	-80(CCB), R7	0635
			00	20	90	002AC		MOVB	#32, @0(R7)	
				67	D6	002B0		INCL	(R7)	
00	B7	50	BE	59	28	002B2		MOV C3	RPT_LENGTH, @REPEAT_DSC+4, @0(R7)	0637
		67		53	D0	002B8		MOVL	R3, (R7)	
		00	B7	28	90	002BB		MOVB	#40, @0(R7)	0638
				67	D6	002BF		INCL	(R7)	
00	B7	04	BE	58	28	002C1		MOV C3	LENGTH1, @LEFT1, @0(R7)	0639
		67		53	D0	002C7		MOVL	R3, (R7)	
		00	B7	2C	90	002CA		MOVB	#44, @0(R7)	0640
				67	D6	002CE		INCL	(R7)	
			50	02	A6	9E 002D0		MOVAB	2(R6), R0	0641
50	D2	AB	10	00	ED	002D4		CMPZV	#0, #16, -46(CCB), R0	
				0C	1E	002DA		BGEQU	39\$	
		7E		42	8F	9A 002DC	38\$:	MOVZBL	#66, -(SP)	0644
		00000000G	00	01	FB	002E0		CALLS	#1, FOR\$\$SIGNAL_STO	
				04	002E7			RET		0643
				0000V	30	002E8	39\$:	BSBW	DO WRITE	0647
			58	B0	AB	9E 002EB		MOVAB	-80(CCB), R8	0648
			00	20	90	002EF		MOVB	#32, @0(R8)	
				68	D6	002F3		INCL	(R8)	
00	B8	00	BE	56	28	002F5		MOV C3	LENGTH2, @LEFT2, @0(R8)	0649
		68		53	D0	002FB		MOVL	R3, (R8)	
		00	B8	29	90	002FE		MOVB	#41, @0(R8)	0650
				68	D6	00302		INCL	(R8)	
				04	00304			RET		0515
				57	D7	00305	40\$:	DECL	R7	0658
		38	AE	57	B0	00307		MOVW	R7, DSC	
		3C	AE	6A	D0	0030B		MOVL	(R10), DSC+4	0659
00A2	09		02	56	CF	0030F		CASEL	LELEM_TYPE, #2, #9	0662
00A2	0048	0048		0057		00313	41\$:	.WORD	46\$-41\$,-	
	0057	0057		0057		0031B			45\$-41\$,-	
		006B		006B		00323			45\$-41\$,-	
									53\$-41\$,-	
									46\$-41\$,-	
									46\$-41\$,-	
									46\$-41\$,-	
									53\$-41\$,-	
									47\$-41\$,-	
									47\$-41\$,-	
	01	1B		56	CF	00327		CASEL	LELEM_TYPE, #27, #1	0691

001A	0004	0032B	42\$:	.WORD	43\$-42\$,-	
					44\$-42\$	
	03	DD	0032F	43\$:	PUSHL	#3
	01	DD	00331		PUSHL	#1
7E	0F	7D	00333		MOVQ	#15, -(SP)
	48	AE	9F	00336	PUSHAB	DSC
	0C	AC	DD	00339	PUSHL	ELEM_ADR
00000000G	00	06	FB	0033C	CALLS	#6, FOR\$CVT_G_TG
		6D	11	00343	BRB	52\$
		04	DD	00345	44\$:	PUSHL
		01	DD	00347		#4
7E		21	7D	00349	PUSHL	#1
	48	AE	9F	0034C	MOVQ	#33, -(SP)
	0C	AC	DD	0034F	PUSHAB	DSC
00000000G	00	06	FB	00352	PUSHL	ELEM_ADR
		57	11	00359	CALLS	#6, FOR\$CVT_H_TG
		38	AE	9F	BRB	52\$
		0C	AC	DD	0035B	45\$:
00000000G	00	02	FB	0035E	PUSHAB	DSC
		48	11	00368	PUSHL	ELEM_ADR
		08	AC	DD	0036A	46\$:
		01	DD	0036D	CALLS	#2, OT\$SCVT_L_TL
		40	AE	9F	BRB	52\$
		0C	AC	DD	0036F	PUSHL
00000000G	00	04	FB	00372		ELEM_SIZE
		34	11	0037C	PUSHL	#1
		0C	AC	D0	0037E	47\$:
30	50	AE	60	D0	00382	PUSHAB
	04	08	AC	D1	00386	DSC
			04	12	0038A	PUSHL
			50	D4	0038C	ELEM_ADR
			04	11	0038E	CALLS
			A0	D0	00390	#4, OT\$SCVT_L_TI
34	50	AE	50	D0	00394	BRB
	0A		01	DD	00398	52\$
			58	D1	0039A	MOVQ
			04	12	0039D	ELEM_ADR, R0
			07	DD	0039F	(R0), D_VALUE
			02	11	003A1	CMPL
			10	DD	003A3	ELEM_SIZE, #4
			40	AE	9F	48\$
			3C	AE	9F	49\$:
00000000G	00		04	FB	003AB	BRB
	05		50	E8	003B2	49\$:
FF70	CB		3F	90	003B5	48\$:
	6A		57	C0	003BA	49\$:
			4C	AE	B5	003BD
				1A	13	003C0
3C	BE	38	AE	20	3B	003C2
				02	12	003C8
				51	D4	003CA
				51	D5	003CC
				0C	13	003CE
				51	C3	003D0
3C	50	6A		50	28	003D4
	BE	61		53	D0	003D9
		6A			04	003DC
						56\$:
						RET

0694
0697
0665
0668
0674
0675
0679
0675
0680
0681
0680
0704
0705
0711
0716
0717
0719
0720
0727

FOR\$\$UDF_WL
1-028

FORTTRAN Write List Directed UDF

N 15
16-Sep-1984 00:52:40
14-Sep-1984 12:32:54

VAX-11 Bliss-32 V4.0-742
[FORRTL.SRC]FORUDFWL.B32;1

Page 20
(4)

; Routine Size: 989 bytes, Routine Base: _FOR\$CODE + 002F

B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
]
^
_
`
~
!
"
#\$
%
&
'
(
)
*
+
,
-
.
/
:
;
=
>
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
]
^
_
`
~
!
"
#\$
%
&
'
(
)
*
+
,
-
.
/
:
;
=
>
?
@
A
B
C
D
E
F
G
H
I

FOR\$\$UDF_WL
1-028

FORTTRAN Write List Directed UDF

B 16
16-Sep-1984 00:52:40
14-Sep-1984 12:32:54

VAX-11 Bliss-32 V4.0-742
[FORRTL.SRC]FORUDFWL.B32;1

Page 21
(5)

```
: 667      0728 1 GLOBAL ROUTINE FOR$$UDF WL9
: 668      0729 1       : JSB_UDF9 NOVA[UE =
: 669      0730 1
: 670      0731 2       BEGIN
: 671      0732 2
: 672      0733 2       EXTERNAL REGISTER
: 673      0734 2       CCB: REF $FOR$CCB_DECL;
: 674      0735 2
: 675      0736 2       JSB_REC9 (FOR$$AA_REC_PR9 + .FOR$$AA_REC_PR9 [.CCB [ISB$B_STTM_TYPE] -
: 676      0737 2       -ISB$K_FORSTTY[0 + -1]);
: 677      0738 2
: 678      0739 1       END;
```

50 FF71 CB 9A 00000 FOR\$\$UDF WL9::

```
50 00000000G0040 D0 00005 MOVZBL -143(CCB), R0
00000000G0040 17 0000D MOVL FOR$$AA_REC_PR9[R0], R0
JMP FOR$$AA_REC_PR9[R0]
```

```
: 0737
: 0736
:
```

; Routine Size: 20 bytes, Routine Base: _FOR\$CODE + 040C

; 679 0740 1


```
: 681      0741 1 ROUTINE DO_WRITE                                ! do per-record formatting and write record
: 682      0742 1 : JSB_REC1 NOVALUE =
: 683      0743 1 | +
: 684      0744 1 | FUNCTIONAL DESCRIPTION:
: 685      0745 1 |
: 686      0746 1 |     DO_WRITE calls the appropriate REC1 level routine depending on the
: 687      0747 1 |     statement type ISB$B_STTM_TYPE.
: 688      0748 1 |
: 689      0749 1 | CALLING SEQUENCE:
: 690      0750 1 |
: 691      0751 1 |     JSB DO_WRITE
: 692      0752 1 |
: 693      0753 1 | FORMAL PARAMETERS:
: 694      0754 1 |
: 695      0755 1 |     NONE
: 696      0756 1 |
: 697      0757 1 | IMPLICIT INPUTS:
: 698      0758 1 |
: 699      0759 1 |     CCB                                Pointer to current logical unit block
: 700      0760 1 |
: 701      0761 1 | IMPLICIT OUTPUTS:
: 702      0762 1 |
: 703      0763 1 |     See module FOR$REC_PROC
: 704      0764 1 |
: 705      0765 1 | --
: 706      0766 1 |
: 707      0767 2 BEGIN
: 708      0768 2
: 709      0769 2 EXTERNAL REGISTER
: 710      0770 2     CCB : REF $FOR$CCB_DECL;
: 711      0771 2
: 712      0772 2 JSB_REC1 (FOR$AA_REC_PR1 + .FOR$AA_REC_PR1 [.CCB [ISB$B_STTM_TYPE] -
: 713      0773 2     ISB$K_FORSTTY[0] + 1]);
: 714      0774 2
: 715      0775 2 RETURN;                                ! Return from DO_WRITE routine
: 716      0776 1 END;                                    ! End of DO_WRITE routine
```

50 FF71 CB 9A 00000 DO_WRITE:

```
50 00000000G0040 D0 00005 MOVZBL -143(CCB), R0
00000000G0040 17 0000D MOVL FOR$AA_REC_PR1[R0], R0
JMP FOR$AA_REC_PR1[R0]
```

: 0773
: 0772
:

; Routine Size: 20 bytes, Routine Base: _FOR\$CODE + 0420

; 717 0777 1

FOR\$\$UDF_WL
1-028

FORTTRAN Write List Directed UDF

D 16
16-Sep-1984 00:52:40
14-Sep-1984 12:32:54

VAX-11 Bliss-32 V4.0-742
[FORRTL.SRC]FORUDFWL.B32;1

Page 23
(7)

: 719 0778 1 END
: 720 0779 0 ELUDOM

PSECT SUMMARY

: Name Bytes Attributes
: _FOR\$CODE 1076 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	20	0	581	00:01.1
_\$255\$DUA28:[FORRTL.OBJ]FORLIB.L32;1	711	190	26	52	00:00.6
_\$255\$DUA28:[FORRTL.OBJ]RTLILIB.L32;1	36	0	0	8	00:00.1

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:FORUDFWL/OBJ=OBJ\$:FORUDFWL MSRC\$:FORUDFWL/UPDATE=(ENH\$:FORUDFWL)

: Size: 1055 code + 21 data bytes
: Run Time: 00:24.5
: Elapsed Time: 01:00.1
: Lines/CPU Min: 1906
: Lexemes/CPU-Min: 16000
: Memory Used: 357 pages
: Compilation Complete

0184 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY